REMARKS

Claims 1-47 are currently pending in the present application. In the Office Action of June 1, 2005, claim "19" was objected to for being incorrectly numbered as claim "18." Additionally, claims 21 and 42 where objected to for failing to distinctly claim the subject matter which Applicant regards as the invention. The Office Action rejected claims 1-2, 4, 7, 17-18, 22-23, 25, 28, and 38-39 under 35 U.S.C. §102(b) as being anticipated by McKeen et al. (U.S. Pat. No. 5,421,022). The Office Action also rejected claims 43-47 under 35 U.S.C. §102(e) as being anticipated by Chaudhry et al. (U.S. Pat. No. 6,684,398). Furthermore, the Office Action rejected claims 5-6, 8, 11, 13-16, 19, 26, 27, 29, 30, 34-37, and 40 under 35 U.S.C. §103(a) as being unpatentable over McKeen et al. in view of Chaudhry et al. The Office Action next rejected claims 3, 12, 20, 24, 33, and 41 under 35 U.S.C. §103(a) as being unpatentable over McKeen et al. in view of Shibayama et al. (U.S. Pat. App. No. 2003/0014602). Finally, Applicant appreciates the Examiner's indication that claims 21 and 42 are allowable.

Claim Objections

The Office Action objected to claim "19" for being incorrectly numbered as a redundant claim "18". Applicant appreciates the Examiner's indication that this typographical error has been corrected. In this regard, Applicant has reflected the amendment in the above-listed claim set.

The Office Action also objected to claims 21 and 42 because the phrase "write instructions within the critical section that do not change a value of memory location being written to" was potentially unclear. That is, the Examiner interpreted the cited phrase as "write instructions that write the same values to memory location." Applicant essentially agrees with the Examiner's interpretation, but would more accurately summarize the cited phrase as "write instructions that *would* write a value to a memory location that is essentially the same as the value stored in the memory location." Specifically, since such write instructions are elided, the write instructions do not actually write but are elided. Therefore, Applicant believes it is more accurate to summarize such write instructions as "write instructions that *would* write...." Nevertheless, as Applicant believes this interpretation does not substantially deviate from the interpretation utilized by the Examiner during examination,

Applicant believes the Examiner's indication of the allowability of claims 21 and 42 should be unchanged.

Rejections under §102(b)

Background

Before addressing the claim rejections in detail with respect to the individual elements of the claims, Applicant believes it would be helpful to highlight some of the fundamental differences between McKeen et al. and the claimed invention. Accordingly, a brief summary of McKeen et al. and the claimed invention follows to serve as an aid when reconsidering the claimed invention and the teachings and suggestions of the prior art of record.

Please note that the following summary of the Detailed Description portion of the Specification is provided in order to facilitate an understanding of the claimed invention with respect to the prior art of record. Thus, the following summary is not to be read so as to limit the claimed invention, but merely to better understand that which is called for in the claims.

Summary of McKeen et al.

McKeen et al. teaches a system for dealing with data dependencies and control dependencies. That is, McKeen et al. teaches a system for dealing with sections of a program that cannot be immediately executed because the data is not yet known (data dependency) and branching decisions that cannot be made because the data is not yet known (control dependency).

McKeen et al. teaches that speculative execution (i.e. execution in the speculative state 2) of grouped instruction sets (i.e. threads) having either data dependencies or control dependencies allows the advantageous reordering of instructions by the compiler with the hope of potentially increasing overall throughput. *See* col. 5, ll. 46-66. During speculative execution, if no exception occurs, data generated during the speculative state "will, at some point in the future, be incorporated into the real state 1." Col. 6, ll. 41-46. However, if an exception occurs in the speculative state, "that exception condition is detected and the set of instructions is re-executed in a real state of the computer system to resolve the exception condition." Abstract and col. 6, ll. 46-49 (emphasis added). Thus, McKeen et al. teaches a system that uses speculative execution to handle two distinct scenarios – (1) unresolved

and/or unresolvable data dependencies and (2) unresolved and/or unresolvable control dependencies.

Summary of the present invention

On the other hand, the present invention deals with a third type of situation that is distinct from (1) data dependencies or (2) control dependencies – (3) thread interruptions. That is, the present invention uses speculative execution to allow multiple threads executing with access to common memory to continue execution through critical sections without restricting (i.e. locking) access to portions of the common memory. In this regard, a given thread is permitted to continue executing through a critical section without setting a lock by speculatively executing the critical section and then committing the speculative execution of the critical section only if there has been no interruption to access of the common memory by another thread. That is, the speculative execution is only committed if, between initial access of a particular portion of the shared memory by the given thread and completion of the speculative execution, there has not been an intervening access to the particular portion of the shared memory by another thread, i.e. an interruption. As such, if there has been an interruption by another thread, the speculative execution of the critical section is squashed.

This process can be exemplified with a common example of using computer threads to query and reserve airline seats. Within this context, a first thread may be used to access a shared memory holding the number of seats remaining for sale on a particular airline flight, for example, a flight having one seat remaining for sale. The first thread accesses the portion of shared memory and determines that it indicates one seat remaining for sale. However, before the thread can reserve the single remaining seat, a second thread reserves the seat. Now, if the first thread were to reserve the seat, the number of seats remaining would be negative one and the plane would be overbooked.

As described in the Specification, this "interruption" of the first thread by the second thread is traditionally averted by the use of locks. However, locks are overly protective by nature and, thus, the present invention allows both threads to execute but squashes one if there has been an "interruption." Extending the present invention to the airline seat example, the first thread speculatively executes a critical section (e.g. checks the seats and reserves a seat). Then, once speculative execution of the critical section is complete (e.g. seat is

speculatively reserved), if the first thread has not experienced an interruption by another thread, the speculative execution is committed. However, if there has been an interruption by another thread (e.g. another thread reserved a seat during the interim), the speculative execution is squashed.

The present invention/McKeen et al.

Therefore, the present invention uses speculative execution to overcome the necessity of overly protective locks associated with the execution of critical sections of a program. That is, the present invention allows threads to execute critical sections concurrently and then squashes speculative execution of a particular thread if there has been an interruption of the particular thread by another thread. On the other hand, McKeen et al. deals with unresolved data dependencies (i.e. execution cannot continue because the data values required to calculate the requested are unknown) and control dependencies (i.e. a branch cannot occur because the data values required to determine the direction of the branch are unknown).

One of ordinary skill in the art will readily recognize that while both techniques employ speculative execution, the speculative execution is performed for different purposes to achieve different ends. In this regard, as will be shown below, McKeen et al. does not teach or suggest the very specific steps called for in the claims.

Claim Elements

The Office Action rejected 1-2, 4, 7, 17-18, 22-23, 25, 28, and 38-39 under §102(b) as being anticipated by McKeen et al. With respect to claim 1, Applicant has amended the claim to clarify the points addressed above. In particular, Applicant has amended the claim to clarify that the criteria for determining whether the speculative execution will be committed is whether an "interruption" of the thread, as described above, has occurred. Applicant believes that the term "interruption" more clearly describes the claimed invention. That is, the term "conflict" carries significant meaning in the art and Applicant believes that, as described above, "interruption" of the given thread more accurately describes the invention. In this same vein, Applicant has been careful to call for an "interruption" as opposed to an "interrupt", which also carries significant meaning in the art.

Similarly, Applicant has amended claim 22 to call for a given thread to access a common memory under the potential for "unpredicted access of the common memory by

other program threads" that "could ***interfere*** with the given program thread." As described above, should a thread "interfere" with the given program thread, the speculative execution performed by the given thread is squashed. However, if there is no such interference, the speculative execution is committed.

Therefore, Applicant believes claims 1 and 22 is patentably distinct from McKeen et al. Furthermore, Applicant believes claims 2-21 and 23-42 are in condition for allowance at least pursuant to the chain of dependency.

Rejections under §102(e)

Applicant does not concede that Chaudhry et al. is valid prior art, and hereby reserves the right to antedate this reference. However, since there are fundamental differences between the claimed invention and the disclosure of Chaudhry et al., in an effort to expedite prosecution of an application that has been pending for nearly four years, Applicant has elected to amend the claims to clarify the claimed invention. These amendments are in no way an admission that Chaudhry et al. is valid prior art under §102(e).

With respect to claim 43, Applicant has amended the claim to clarify that "wherein upon complete execution of the critical section without acquiring permission to write to the lock variable, if another thread has acquired the lock, squashing the execution of the critical section and then re-executing the critical section." Chaudhry et al. clearly does not teach or suggest that which is called for in claim 43.

Specifically, Chaudhry et al. teaches, "During a monitor exit, speculative thread 203 decrements the number of virtual locks 1218 (step 1402)." Col. 10, ll. 20-22. "Next, if the number of virtual locks equals zero, and if no locks in the list of virtual locks are currently held by other head threads, speculative thread 203 commences a join operation with head thread 202, if such a join operation is pending (step 1404)." Col. 10, ll. 22-26. Nowhere does Chaudhry et al. teach or suggest that "upon complete execution of the critical section without acquiring permission to write to the lock variable, if another thread has acquired the lock, squashing the execution of the critical section and then re-executing the critical section." In fact, such an event would not arise under the technique taught by Chaudhry et al. because Chaudhry et al. teaches that the speculative thread 203 will not allow a joint operation to commence while a virtual or non-virtual lock is in place. *See* col. 10, ll. 22-26.

For at least these reasons, claim 43 is patentably distinct from Chaudhry et al. Additionally, claims 44-47 are in condition for allowance at least pursuant to the chain of dependency. It should be noted that claims 45 and 47 have been amended only to improve consistency with claim 43 after amendment. The amendments to claims 45 and 47 were not made to overcome the prior art rejections.

Rejections under §103(a)

The Examiner rejected claims 3, 5-6, 8, 11-16, 19, 20, 24, 26, 27, 29, 30, 33-37, 40, and 41 under 35 U.S.C. §103(a) as being unpatentable over various combinations of McKeen et al., Chaudhry et al., and Shibayama et al. Applicant respectfully disagrees with the Examiner with respect to the art as applied, but in light of claims 3, 5-6, 8, 11-16, 19, 20, 24, 26, 27, 29, 30, 33-37, 40, and 41 depending from what are believed otherwise allowable claims, Applicant does not believe additional remarks are necessary and therefore requests allowance of claims 3, 5-6, 8, 11-16, 19, 20, 24, 26, 27, 29, 30, 33-37, 40, and 41 at least pursuant to the chain of dependency.

Conclusion

Applicant respectfully asserts that all rejections and objections asserted in the Office Action have been overcome. In particular, in light of the foregoing, Applicant believes the previous bases of rejection presented against claims 1-47 have been overcome. Accordingly, Applicant believes the application is in condition for allowance and a Notice of Allowance is respectfully requested. However, should the Examiner believe any matter remains outstanding, the Examiner is invited to contact the undersigned at the telephone number appearing below if such would advance the prosecution of this application.

Respectfully submitted,

RAVI RAJWAR et al.

By: _____

Jack M. Cook
Reg. No. 56,098
Attorney for Applicant
Quarles & Brady LLP
411 E. Wisconsin Avenue, 20th Floor
Milwaukee WI 53202-4497
(414) 277-5405